LA-UR 92- 67

LA-UR--92-67

DE92 007718

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE USING OBJECT ORIENTED ANALYSIS & DESIGN TO STUDY THE SSCL SDC COMPUTING SYSTEM

AUTHOR(S) Glenn T. Kubena (IBM), Andrea P.T. Palounek (LANL, P-2), Chris Day (LBL), Ken Liao (IBM)

SUBMITTED TO Second International Workshop on Software Engineering, AI, and Expert Systems For High Energy and Nuclear Physics, January 13-18, 1992

MASTER

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Using Object Oriented Analysis & Design To Study the SSCL SDC Computing System

Second International Workshop on Software Engineering, AI, and Expert Systems For High Energy and Nuclear Physics January 13-18, 1992

Glenn T. Kubena (IBM), Andrea P.T. Palounek (LANL)
Chris Day (LBL), Ken Liao (IBM), et. al.

Point of Contact: Glenn Kubena
IBM Federal Sector Division
Houston, Texas
(713) 282-7635
KUBENA@HOUVMSCC.VNET.IBM.COM

SSCL/SDC

DISCLAIMER

Contents

1. Abstract

A joint study between the Computer Working Group of the SSC Solenoidal Detector Collaboration (SDC) and IBM's Federal Sector Division is focusing on the use of Object Oriented Analysis and Design on the SDC Offline Computing System. One key challenge of the analysis is to discover an efficient way to store and subsequently retrieve raw and reconstructed event data, estimated to be 2 petabytes per year.

The Object Oriented approach being used during the analysis and early design is intended to yield a smooth transition to detailed design, prototyping and implementation. The object oriented approach is used as a subprocess of a larger process used by IBM FSD, i.e., a systematic approach to architecting and integrating large complex systems. A description of the overall process and early results are described in a study report (see reference 1) produced jointly by the SDC and IBM FSD. The overall process focuses on requirements analysis, operational concept development, problem domain decomposition, development and selection of candidate architectures, automated performance modeling and software architecture. This paper will focus primarily on software architecture.

The high level software architecture is viewed as a layered stack consisting of: system services, common physics application framework and unique physics applications. Object oriented analysis is being used to investigate the data storage and management of the event data. An object hierarchy is being created and operational concept scenarios are being used to validate the design. Several data base prototypes can then be developed, e.g. object oriented or relational, to prove the concept.

The object oriented development is fundamentally different from traditional functional approaches to design, such as those based exclusively on data flow. Object oriented decomposition more closely models a person's perception of reality, hence the developed system is more understandable, extensible, and maintainable.

Although the description of the above process is of necessity linear, the actual development process is iterative. The object oriented methodology makes it easier to repeat the development steps at progressively finer levels of detail.

2. Introduction

2.1 Background on IBM FSD

IBM Federal Sector Division has about 13,000 employees and annual revenues of over $3 billion. IBM FSD has 6 sites in the United States including Bethesda, Owego, Gaithersburg, Manassas, Boulder and Houston. IBM FSD Houston has a 30 year legacy supporting NASA Johnson Spaceflight Center from the Mercury program through Gemini, Apollo, Skylab, ASTP, Space Shuttle and the current Space Station program.

IBM FSD Houston has an experienced skill base in large complex systems in software engineering, systems integration, and program management. FSD Houston also has lab facilities for advanced technology, prototyping new systems and live test demonstrations. The IBM FSD Houston shuttle flight software project was evaluated by NASA at level 5 (highest level) on the Carnegie Mellon Software Engineering Institute Evaluation Scale in 1989. IBM FSD Houston also won the award for best software lab within IBM in 1990 and 1991.

2.2 SDC Computing Architecture Study

The SDC Computing Architecture Study (see reference 1) was a joint study between members of the Solenoidal Detector Collaboration and IBM FSD. The study covered the period from April to December, 1991. The study focused on early analysis & design of the SDC Offline Computing System. The study was also used to support the SDC Proposal to the Department of Energy, which is to be delivered in April of 1992. The study includes the following topics.

- Operational Concepts Development
- SDC System Architecture Analysis
- SDC Software Architecture Analysis
- Performance Modeling
- Standards and Technology Forecast
- Costing

This paper will focus on the results of system and software architecture analysis.

3. SDC System Architecture Analysis

3.1 Process Used

The process being used is one suitable for the analysis phase of the system life cycle, i.e., prior to the start of software and hardware design.

The System Architecture Analysis process, as portrayed in figure 1, consists of the following steps. First, requirements from the SDC Proposal Computing Section draft were analyzed and sorted into requirements which addressed each of the following categories:

- System Boundary
- External Interfaces
- Major Functions
- Major Data Sets
- Internal Interfaces

The above was used to produce a Problem Domain Block Diagram which is described below under the results section.

A Operational Concept was developed by an IBM engineer gathering and iterating upon written scenarios from SDC physicists who assumed the role of end users. The details of this Operation Concept are provided in the study (reference 1). The Operational Concept gives valuable insight into how the end users expect the system to be used and is a valuable tool in subsequent validation of proposed architectures.

Evaluation criteria reflecting cost, schedule, performance and environment were developed next via extraction from the proposal requirements. These were then reviewed with SDC members. Evaluation criteria are used to pick the best choice of candidate architectures later in the process. Evaluation criteria are identified early in the process in order to make the final selection more objective.

After the Problem Domain Block Diagram was developed, several target architectures were identified. This is basically as far as the study has gone to date, relative to analysis of the system architecture. The next step will be to use the evaluation criteria to select a best candidate(s). Finally, the selected architecture will be verified through reviews and applying the operational concept. At that point there is feedback into the requirements and architectural analysis and the whole process iterates.

SDC System Architecture Analysis Process Used

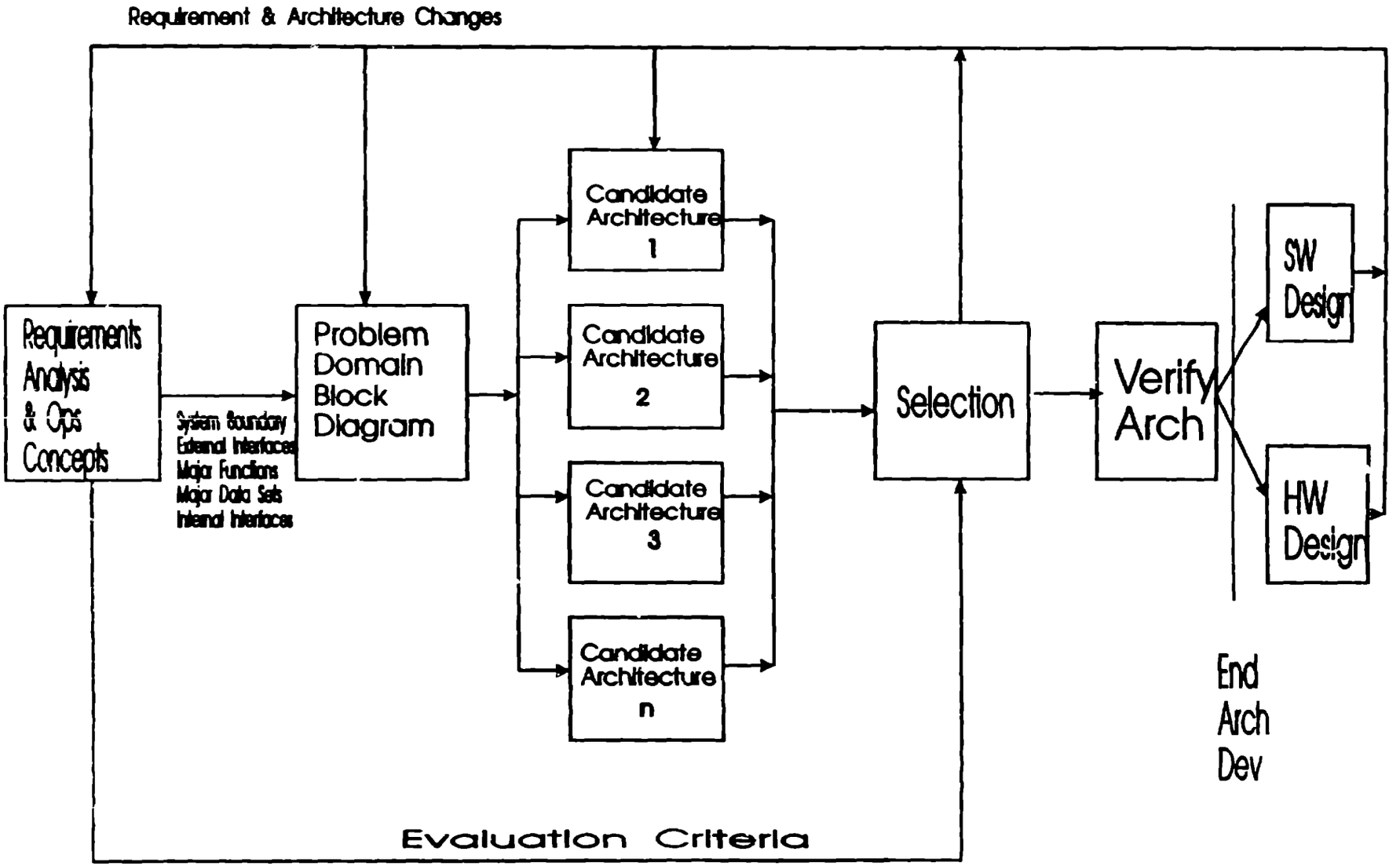


Figure 1

3.2 Results of SDC System Architecture Analysis

3.2.1 Evaluation Criteria for SDC Offline Computing

The following evaluation criteria were identified as selection criteria to judge alternative proposed architectures.

- Cost
  - Hardware acquisition & maintenance costs
  - Software acquisition & maintenance costs (re-use, make or buy options)
    - Public domain (HEPLib, etc.)
    - Custom developed internally or with outside help
    - Commercial software distributed licences & maintenance fees
- Schedule Risk
  - Ability to meet all critical milestones
  - Availability of products to meet evolving standards
  - External dependency risks
  - Availability of critical skills
- Performance
  - 100 raw events/second, received from level 3 processing and processed through production reconstruction without loss of data.
  - Storage of 100 reconstructed events/second or 10**15 bytes/year (event size = 1 megabyte; 10**7 seconds/year). An equal amount of storage assumed for raw event data.
  - Any two of three activities supported concurrently (production reconstruction, second pass reconstruction, simulation)
  - Data reduction of the 10**15 bytes of reconstructed data to a 10**14 byte archive sample and a 10**9 event online sample.
  - Transfer of a 10**5 event sample extracted from from the 10**9 event online sample to a single user's workstation within 3 hours.
  - Access to any event in the raw or reconstructed archive within 24 hours. Access to a large number of archived events within 2 weeks.
  - Accommodate 200 concurrent analysis users including 50 at the SSCL and the remainder at regional centers/institutions.
- Environment
  - Technology insertion and software portability
  - Common user interface; support for expert and ad hoc users
  - Data structures transparent to physics analysis users
  - Interoperability between the SSCL and remote institutions
  - Levels of code management across distributed environment, e.g.:
    - Production reconstruction at the SSCL
    - Analysis at remote institutions
  - Automated test tools and test data generation

3.2.2 Problem Domain Block Diagram

Analysis of the SDC Proposal Computing Section requirements resulted in the Problem Domain Block Diagram in Figure 2. The term "Problem Domain" means that it is an attempt at problem definition or "what" the system is intended to do (as opposed to "how" the system will be designed or built). The Problem Domain block diagram is then used as a basis for exploring candidate architectures.

The Problem Domain Block Diagram portrays an SDC "system boundary" as indicated by the bold line with external interfaces" overlaying the system boundary. External interfaces are:

- The Level 1 and Level 2 part of Data Acquisition which feeds Level 2 data to the Level 3 part of Data Acquisition

- Detector Control Devices which output environment performance data and receive control signals. These interfaces are with the Detector Control & Monitoring functional component.
- Detector Operators who enter control requests and receive performance data from the Detector Control & Monitoring functional component.
- Physics Users who have a number of interfaces as shown with Data Storage & Management, Simulation and Analysis functional components.

Functional Components are:

- Data Acquisition Level 3 triggering which provides Level 3 data to Data Production & Classification and to Data Storage & Management. Data Acquisition Level 3 Triggering receives control signals from Detector Operators via Detector Control & Monitoring.
- Data Production & Classification receives Level 3 data from:
  - Data Acquisition (realtime level 3 data)
  - Data Storage (playback of real level 3 data)
  - Data Storage (playback of simulated level 3 data)

  Data Production & Classification receives control signals from Detector Operators via Detector Control & Monitoring.
- Data Storage & Management receives data (for storage) from a number of sources as shown on the block diagram. Data Storage & Management in turn provides data to a number of destinations as shown.
- Simulation interacts with Physics Users and provides simulated data to Data Storage & Management and to the Analysis functional component.
- The Analysis functional component provides analysis results to Physics Users in response to their request criteria. Analysis results can also be stored via Data Storage & Management. Inputs to analysis are reconstructed data, prior analysis data, and (more rarely) level 3 data from Data Storage & Management. Simulated reconstructed data or level 3 data is also available from the Simulation functional component.

Finally a number of major categories of data are encapsulated (owned) by the the Data Management & Storage functional component. These data categories are listed in the cylinder icons on the figure.

Problem Domain Block Diagram

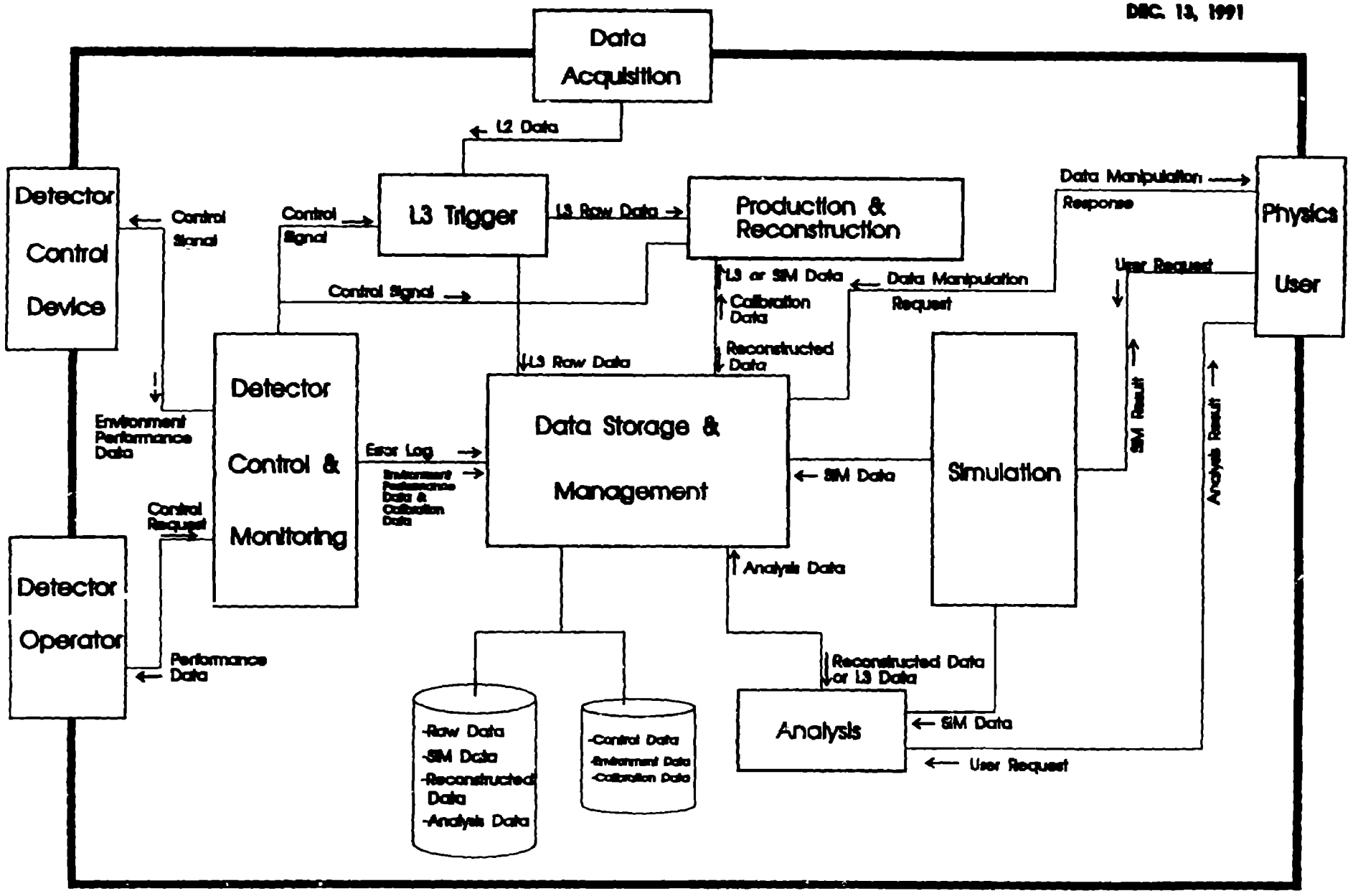


Figure 2

3.2.3 SDC Conceptual Architecture

Computing technologies have advanced significantly in the past five years and should continue to do so. Certainly the number of companies associated with the computing and communication industries have exploded in the past ten years. Today, many vendors are specializing in particular areas of computing which has lead to several developments in the industry.

- More distributed solutions, due to picking the "best of breed" products from a variety of vendors.
- Need for standards to allow connectivity between this variety of products.
- Need for standards to allow technology insertion as the products continue to evolve.
- Increased system performance via product "specialities".

We have all witnessed the decline of the general purpose, large mainframe computing solution in favor of the less costly smaller distributed systems. Before, where computing meant a mainframe, one can now specialize and choose from powerful workstations, computing clusters, massively parallel computers, vector supercomputers, as well as other symmetric multi-processors (from workstation to mainframe class). All in a variety of sizes, prices and from a variety of vendors. Similarly, advances have occurred in communications technologies (Ethernet replaced by gigabit LANS and switches), and storage technologies (optical disk, helical scan tape, robotic libraries, etc.). These advances have lead to a building block approach for cost effective large computing systems.

Future generations of computing (processing and data storage) systems will consist of a collection of such high performance "blocks" or servers made up of computing engines, database engines, and storage subsystems, all connected via networks and managed by a network file server and system manager. Local data will be transferred directly between storage subsystems and computing engines via a switched point-to-point network fabric, obviating the traditional need for store and forward through an intermediate computer.

The SDC Conceptual Architecture, portrayed in Figure 3, represents an architecture which maps to emerging industry standards for such a high performance data system. This generic model features multiple compute servers, data storage servers, and system management servers coupled together by a high speed data network and a low speed control network. This type of architecture offers the following advantages:

- Separation of communications for data and control. The higher cost systems interconnect is connected to those system components which need the higher bandwidth data paths, while separate lower cost control paths prevents control bottlenecks with bulk data flow.
- Concurrent high speed data storage and retrieval. For example, data from the Reconstruction Processor Farms can be stored on the Hierarchical Disk and Archival Farms at the same time that the Analysis Workstations are retrieving data files. This data movement could be accomplished via an intelligent HIPPI switch (100 Mbyte/sec), or a future option such as the Fibre Channel.
- Functionally coherent servers which will exploit present & future standard components, e.g., ISO systems management or IEEE Mass Storage Reference Model.
- Scalable servers, communications and storage subsystems. Each server is tailored in size and cost to best match the user requirements. For example, the Reconstruction Processor Farm may be several independent clusters of tightly coupled workstations attached to the network (scaled to meet current processing needs), yet easily expanded via additional clusters to meet future processing needs (growth).
- Automated hierarchical data management, with migration and caching. For example, the hierarchical storage may consist of multiple Disk Arrays, an Optical Jukebox and multiple automated helical scan tape libraries as independent subsystems all attached to the network, yet managed via the network file server as a single integrated system
- Maximizes use of Commercial-Off-The-Shelf (COTS) hardware and software thereby reducing life cycle cost (lower maintenance, upgrade-ability, etc.).

The SDC conceptual architecture, as portrayed in Figure 3, is a framework for building different hardware architectures by selecting different option. for the boxes in the diagram, e.g.:

- Disk Farm --
  1. High speed disk arrays.
  2. Distributed disks.
- Archive Libraries --
  1. Helical Scan tape (Metrum, E-System, Sony, Exabyte).
  2. Future Optical Tape Libraries (LaserTape, Creo).
  3. Optical Disk Jukebox (Kodak, LSMI, HP).
- Reconstruction Proce: Farm --
  1. Tightly or loosely coupled clusters.
  2. Massively parallel processors.
- High speed system interconnections --
  1. Fibre channel standard.
  2. HIPPI.
  3. FDDI.

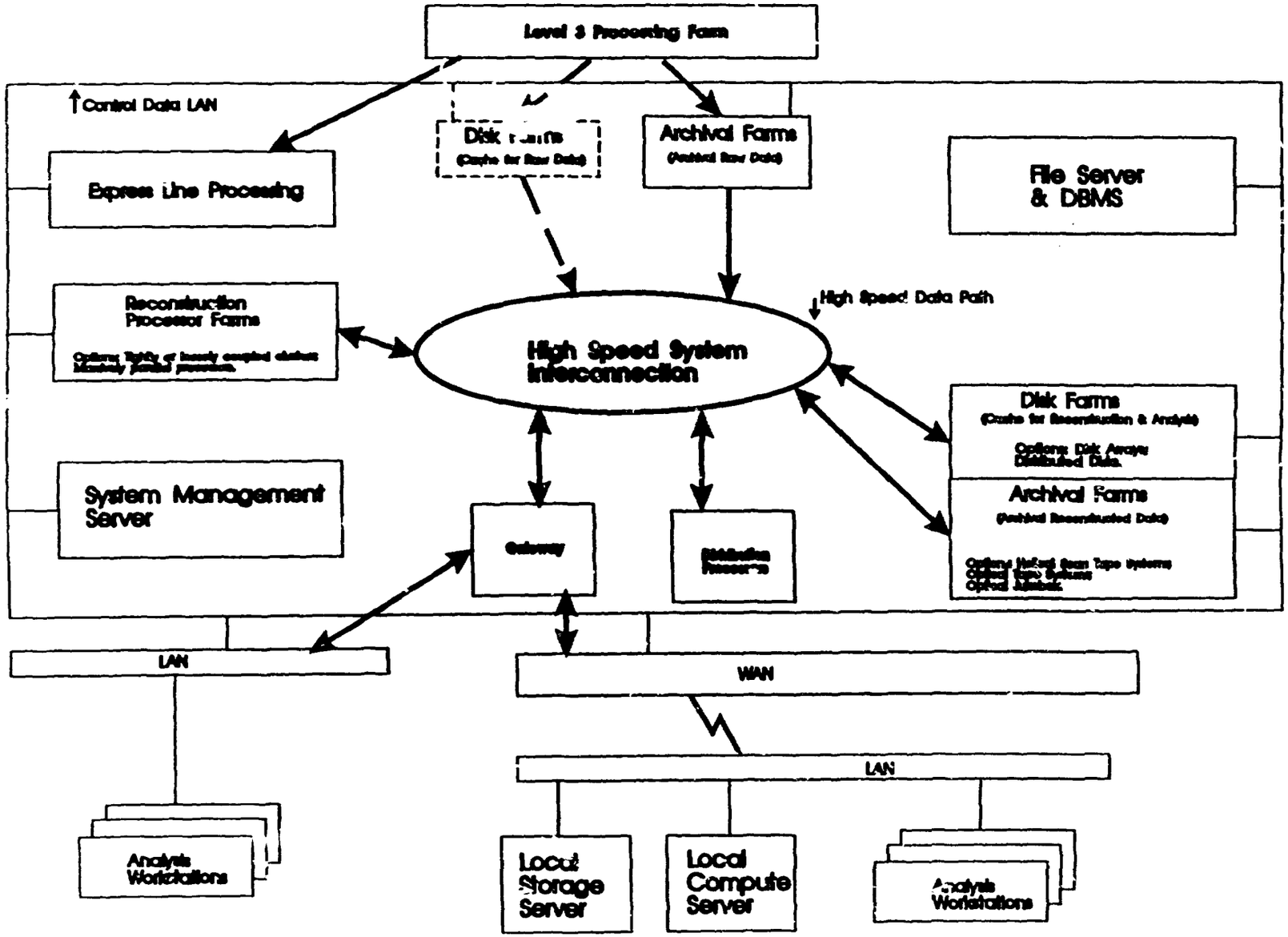
Different options of hardware architectures are developed using the following assumptions:

- A 10 second latency in production reconstruction is acceptable as long as no level 3 data is lost, e.g., to meet the 100K Mips needed, one could use one thousand 100 Mip processors where each processor would take 10 seconds to process a single event.
- Analysis computing resources for 50 local SSCL users must be provided. A typical analyst would require 2 to 3 jobs turned around during prime shift and more overnight.
- A typical analysis file transfer for a single user will be about 10 Gigabytes (10**5 reconstructed events @ 0.1 Mbytes/event). Assuming a 1 Mbyte/second tranfer rate this data can be transferred to the user's workstation in about 10**4 seconds or 3 hours.
- Production reconstruction of event data is highly suited for parallel processing because each event can be reconstructed totally independent of the other events.

The architecture study (reference 1) explores in some detail three major hardware options and three sub-options within option one. However, for purposes of exploring software architecture options it is sufficient to start with the SDC Conceptual Architecture from Figure 3.

SDC Conceptual Architecture Diagram

October 24, 1991



4. SDC Software Architecture Analysis

4.1 Process Used

The process consists of the following steps:

1. Identified key Software Architecture Design Goals. These are listed below under results. It is important to establish goals early in the process in order to focus the architecture towards meeting the goals. Also, establishing goals early in the analysis helps to focus on a better definition of the problem to be solved and can force end users to state their objectives about critical needs and expectations.

2. Proposed a conceptual Software Architecture Model consisting of three layers. This type of model is representative of the ISO/OSI type of model and is particularly useful in distributed environments. The model is shown in detail in the results section below. Drivers for the three layers are:
   - System Services Layer - Driven by portability, interoperability, reliability, common user interface and data transparency requirements.
   - Common Physics Applications Layer - Driven by the need for a common framework for physics reconstruction, analysis, simulation and test.
   - Unique Physics Applications Layer - Driven by the need for easy analysis software development by end users within the common framework.
3. Proposed a conceptual Physics Analysis Framework.
4. Performed Object Oriented Analysis on Data Storage Management.
   - Produced Event Topology Model.

4.2 Results of SDC Software Architecture Analysis

4.2.1 Software Architecture Design Goals

The following software architecture design goals were identified.

- Software should be maintainable, efficient, reliable, understandable, reusable and extensible.
- Software portability and interoperability should support technology upgrades and multi-vendor distributed environments. The SSCL must provide for standard interoperability with remote institutions. Source code portability of the analysis software environment should be provided for all collaborating institutions.
- The communications software should provide for easy exchange of mail, messages and graphical data between external users.
- The user interface with the software should be standard and easy to use. The user interface should support ad hoc as well as expert users. The user interface should evolve to support point-and-click analysis.
- The software architecture should provide for a software bus or infrastructure of standard services utilizing industry standard components wherever possible. The goal is to incur the cost of common services software only once and to promote system integrity.
- The software architecture should provide an environment or framework which makes it easy for physicists to add or modify analysis code without rewrite of the framework. Access to data by such analysis

code should be via an easy to use object query language, i.e., knowledge of internal data structures should be transparent to physicists developing routine analysis code.

- The software architecture should include tools for test data generation and validation against predetermined results or test scripts.
- The software architecture should provide a reasonable level of online help, error recovery and error diagnostics.
- The software architecture shall meet all performance requirements.
- Software shall be designed within budget and within schedule.

4.2.2 Software Architecture Model

The SDC Software Architecture Model is portrayed as a three layer stack in Figure 4. The bottom layer of the stack is System Services. System Services provide the base services which the user normally does not see or views collectively as "operating system". The "user" in this context is anyone developing code for the middle layer or in some cases an interactive end user. Systems Services meet the goals of providing system reliability, interoperability, portability, transparency of internal data structures and common user interface System Services exploits the use of current and emerging industry standards such as ISO/OSI, POSIX, X-Windows, etc. For a detailed discussion of standards please see section 6.1 of the study report (reference 1).

The middle layer of the stack is Common Physics Application Services. This layer provides for the day-to-day control, monitoring and production of reconstructed physics events. It also provides the framework which supports physics analysis. Simulation is included in this layer. Finally, the software development & testing environment is included.

The third layer contains unique physics application code, e.g., user written analysis routines.

Figure 4 also shows an expanded view of the software architecture model. System services are broken down into more detail and API's (application program interface's) are shown. API's are the externally visible part of the services, i.e., that part of the service as seen by the software user of it. For example, Data Reconstruction & Classification may use the Network I/O API to send data over a LAN. The software user knows how to use a service via the API, the implementation details are hidden. This type of model can lead to a distributed design where API's may be local or remote as in a client server model.

Software Architecture Model

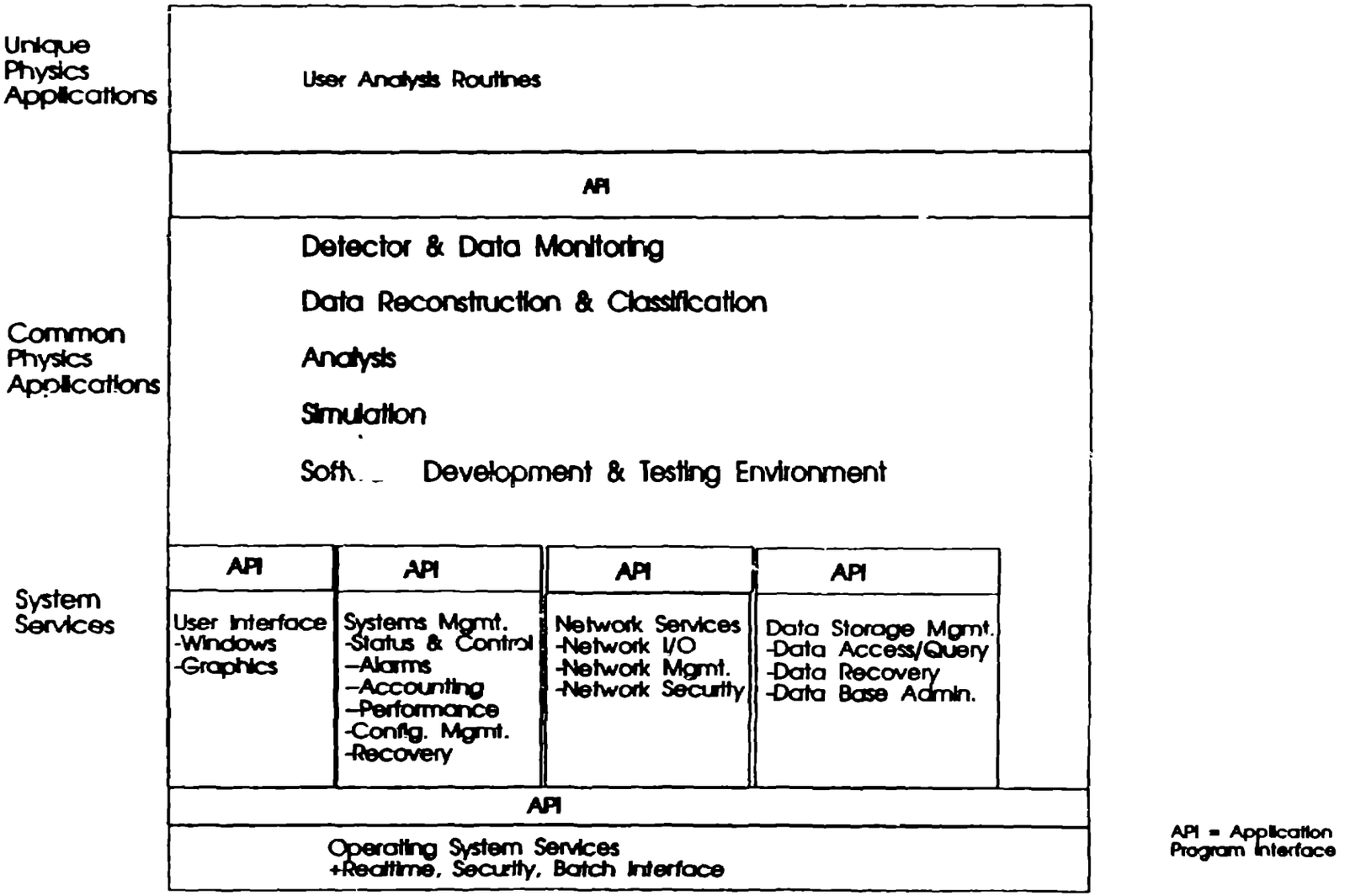


Figure 4

4.2.3 Analysis & Simulation Software Architecture

The Analysis & Simulation Software Architecture is portrayed in figure 5. The architecture assumes the use of a DBMS (DataBase Management System). Common physics analysis and simulation codes are written in modules which could be linked together by the job setup routine without the need of recompiling. The Job Setup GUI (Graphical User Interface) is the point-and-click user interface which allows the user to link the common physics analysis or simulation routines together to accomplish an analysis or simulation task. There are two RuleBases in this architecture. The Query RuleBase is used as a supervisory agent for the interaction between DBMS and the rest of the system, and the Framework RuleBase is used to store the rules which assure the compatibility of the two modules for linking.

A typical user scenario may look like the following:

- User places icons (a graphical representation of physics modules) on the screen and specifies the parameters for every module.
- User connects modules by point-and-click.
- In turn, the system checks each connection, as it is made, against the Framework RuleBase to verify that the connection is possible.
- If the connection is illegal, the user is warned and has to redo the connection by specifying different parameters or using different modules.
- Once the setup is done, the user could submit the job as a batch run or run it interactively.
- When the physics module access the DBMS (read or write), the query is checked against the Query RuleBase to make sure the query is a legitimate one. For example, a user makes a query that may cause a terabyte of data to be sent. The RuleBase then may issue a warning message to the user. Or, a user make a query to the DBMS which violates some physics rules. The RuleBase could reject such a query.
- If the query passes the check then it is passed to the DBMS for processing, otherwise a warning message is returned back to the user and the query is aborted.

This architecture facilitates point-and-click physics analysis and simulation. Physics modules are represented on the screen as icons. Users can specify an initial environment for a module by double clicking on an icon to bring up a setup dialog box. Modules could be connected together to establish inter-communication by drawing a line from a output port of one module to a input port of another module provided the connection complies to the rules in the Framework RuleBase.

The use of object oriented methodology encapsulates the data structure for the physics analysis and simulation modules, hence a higher maintainability of the codes. Message passing mechanism provides an easier and more flexible way to integrate the actual physics code with the graphical user interface. Together with the RuleBase system, a highly extensible and maintainable framework could be established. As the physics experiments advanced, new pieces of framework can be easily added or the old one can be easily updated. The RuleBase provides the mechanism to record the intercommunication restrictions among different modules without hard-coding them in the program, since the rules are most volatile part of the framework. Updating these restrictions requires only updating the RuleBase without the need for re-compiling and re-linking the program.

Analysis & Simulation Software Architecture

Dec. 10, 1990

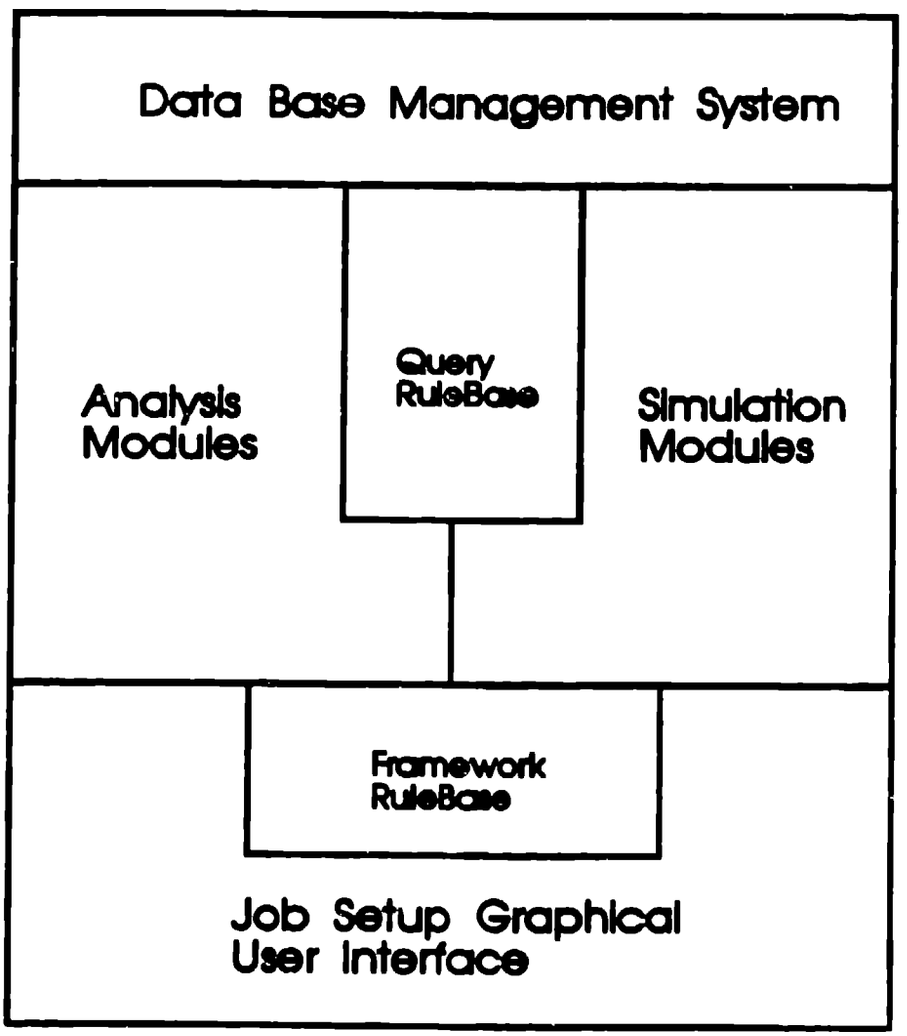


Figure 5

4.2.4 Event Topology Model

4.2.4.1 Overview

The Event Topology Model is the resulting model produced by applying Object Oriented Analysis to the problem domain of the Data Storage Management Subsystem. The model can be expanded to become a full scale data base schema during the design phase of the Data Management Subsystem.

The Data Storage Management subsystem is a key component of the system because it has to effectively manage the massive amount of data and provide timely delivery to the end-users. Simplicity, flexibility and expandability are the key drivers for the OOA model during this analysis and throughout the Storage Management Subsystem design phase.

An explanation of the method and notation used in the analysis is in order before proceeding to the analysis results. Section 6 of this paper gives references to methodology books and papers used. The tool, OOATool(TM), which follows the notation used in the Peter Coad book was used to develop the model. The notation used by OOATool is portrayed in Figure 6. Also, the book by Rumbaugh, et.al., was heavily used as a reference, even though the Rumbaugh notation is different than OOATool. Figure 6 shows the notation for portraying a Class (a collection of objects with common attribute names,types and services) and a Class & Object (where an object is an individual instance of the class attribute values and services). There are two major inter-class relationships which interconnect and amalgamate classes into a hierarchy, the "Whole-Part" and the "Generalization-Specification" relationships. Non-hierarchical relationships are portrayed via the "instance connection". The "Whole-Part" relationship is shown with a whole class at the top, and then a part class below, with a line drawn between them. A triangle annotation distinguishes classes as forming a Whole-Part relationship. Each end of a Whole-Part relationship line is marked with a cardinality annotation, indicating the number of parts (e.g. range 1,m) of a given kind associated with the whole. On the other hand, the "Generalization-Specialization" relationship is shown with a Generalization Class at the top and Specialization Class below, with lines drawn between them. A semi-circle annotation distinguished Classes forming Generalization-Specialization relationships. The basic way to distinguish Whole-Part from Generalization-Specialization is as follows. The parts of a Whole-Part form an "and relationship". For example, Figure 7, shows that a microcomputer parts are a monitor "and" a system box "and" a mouse "and" a keyboard. A Generalization-Specialization is an "is-a" hierarchy where the subclasses form an "or relationship". For example in the figure, a worker "is a" butcher "or" a baker "or" a candlestick maker. Further, for Gen-Spec, attributes and services are inherited by the subclasses as in Figure 7. The attributes and services can be overridden by the subclass and new attributes and services can be added.

Figure 6

Object Oriented Analysis Tool (TM) Notation

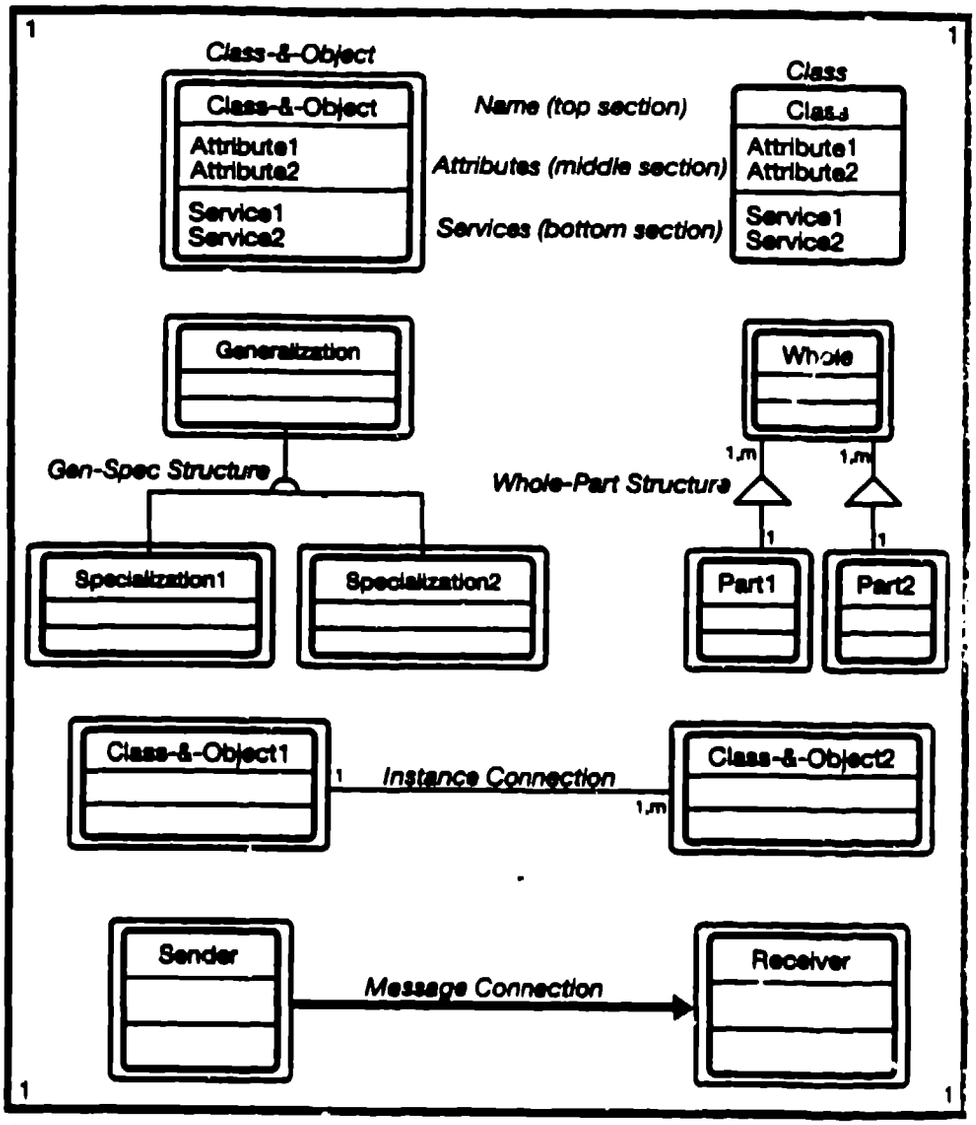
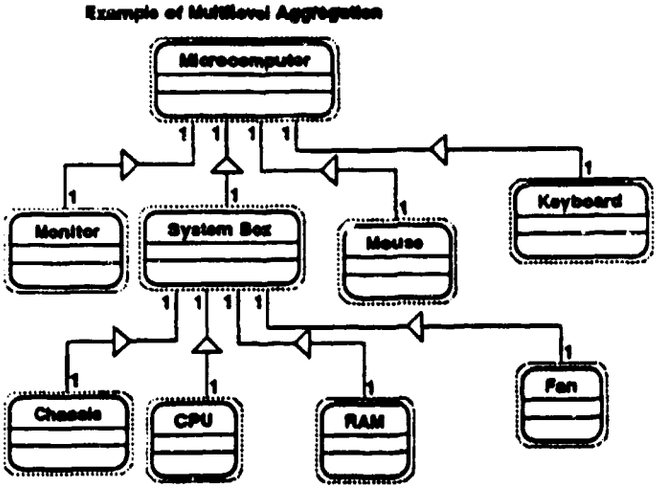
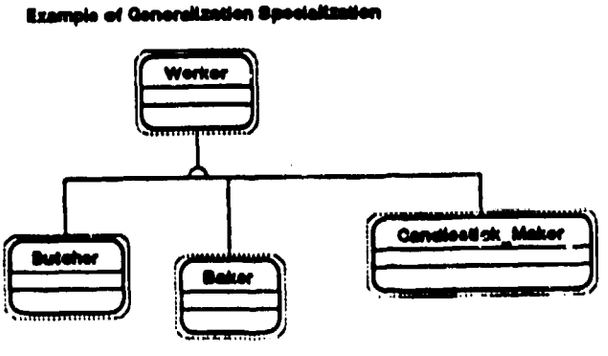


Figure 7


Example of Multilevel Aggregation
Microcomputer
Monitor
System Box
Mouse
Keyboard
Chassis
CPU
RAM
Fan


Example of Generalization Specialization
Worker
Butcher
Baker
Candlestick_Maker

4.2.4.2 Analysis Results: Candidate Model

The candidate model, portrayed in Figure 8, is the result of initial analysis. It is important to note that this model will continue to evolve as the problem domain is better understood and physics analysis needs evolve. For example, the model currently shows no generalization-specialization (superclass-subclass) relationships. Generalizations may be added later as abstract classes are recognized. Specializations may be added later to provide unique attributes and/or services without disturbing the existing structure. Other aspects of the model may change as the result of early prototyping efforts.

A description of the model classes and their relationships, for the model as it currently stands, are given below.

4.2.4.2.1.1 Event Class: The Event class contains object instances of "events" or collisions which have passed through the level 3 trigger. The Event class is the smallest unit which distinguish one collision from the others. The Event class contains attributes such as Event_Number, Run_Number, Trigger_Number, etc. to uniquely identify itself. The event class is modeled as an aggregate. The "parts" of the aggregate are Cluster "and" Vertex "and" Track "and" Candidate_Particle. The Event class objects are "instantiated", i.e., the attribute values are stored, as part of production reconstruction processing. Some of the attributes may be instantiated or updated as part of subsequent physics analysis processing. In fact, all of the event topology classes (Cluster, etc.) are instantiated by production reconstruction and physics analysis processing.

Two key attributes present in many of the classes are Algorithm_Version and Name. This allows for the fact that the same Event and Event parts may be classified in different ways depending upon the algorithm used. For example, there may be several Candidate_Particle object instances which are part of the same Event object instance, as follows.

- Candidate_Particle Object 1
  - Candidate_Particle_Type = Muon
  - Probability = 50%
  - Algorithm_Name = XYZ
  - Algorithm_Version = 1
- Candidate_Particle Object 2
  - Candidate_Particle_Type = Electron
  - Probability = 50%
  - Algorithm_Name = XYZ
  - Algorithm_Version = 2

4.2.4.2.1.2 Parts of an Event Aggregate: The "parts" of the event aggregate are Cluster "and" Vertex "and" Track "and" Candidate_Particle.

- Cluster

  The Cluster class is part of the Event class and is itself an aggregate. The parts of Cluster class are the EM_Cluster and Hadronic_Cluster classes. This class aggregate is used to classify data which is produced by the detector calorimeter. The attributes are energy deposit values and location information, e.g., Total Energy (sum of EM_cluster energy and Hadronic_Cluster Energy), Algorithm_Name and Version used in reconstruction, Shape (shape of the cluster) and Position (position vector of the cluster). The EM_Cluster and Hadronic_Cluster classes are parts of the Cluster class because the EM_Cluster is the energy detected by the Strip Chamber while the Hadronic_Cluster is the energy detected by the outer part of the Calorimeter, together they make up the Cluster.

- Track

The Track class contains reconstructed tracking information. A track class is an aggregate of tracking segments from the tracking detectors: silicon, inner, outer and forward. The Track class has attributes such as Algorithm_Name, Algorithm_Version, Tower_Location (the Calorimeter tower address which the track is pointing to), Position (position vector of the track), Momentum (momentum vector of the track) and Error (accuracy).

- Vertex

  The Vertex class records the point at which track(s) originate and in some cases end. There are two kind of vertices. The primary and the secondary vertex. The primary one is created by a particle collision while the secondary is created by particle decay or secondary interactions in the detector. The Vertex class has the following attributes: XYZ (coordinate in space), Incoming_Track_Number (the track which comes into the vertex, this would be zero if this is a primary vertex) and List_of_Outgoing_Track_Numbers (the track(s) which expand out from this vertex). The Vertex Class is associated with the Track Class as follows. A Vertex object may be associated with 0 or 1 incoming tracks and 1 to N outgoing tracks.

- Candidate_Particle

  The Candidate_Particle class contains candidate_particle objects of various candidate_particle_types, e.g., electron, photon, muon, jet, neutrino, ... The other attributes are probability (of the particle_type) and algorithm_name and version. An initial step of classifying particles by particle type is performed by production reconstruction processing according to the table in Figure 4.2.1-5. Notice that Candidate_Particle has an associative relationship with Cluster, Vertex and Track in order to do this classification. That is, for each Candidate_Particle object there is 0 or 1 Cluster object (A zero Cluster object would occur for a neutrino). There could be 0 or 1 Vertex object and 0 or 1 Track object associated with the Candidate_Particle object.

4.2.4.2.1.3 Environment_Status Class: The Environment_Status class is used to keep environmental data associated with the Event class correlated by time. For N events there is an instance of environment status collected over a given time period, i.e., while Events occur every 16 nanoseconds, the environmental data may be collected once a second or slower. The service Store/Fetch_Environ_Status_by_Time would allow the user or software to access the environment attributes for related events by time.

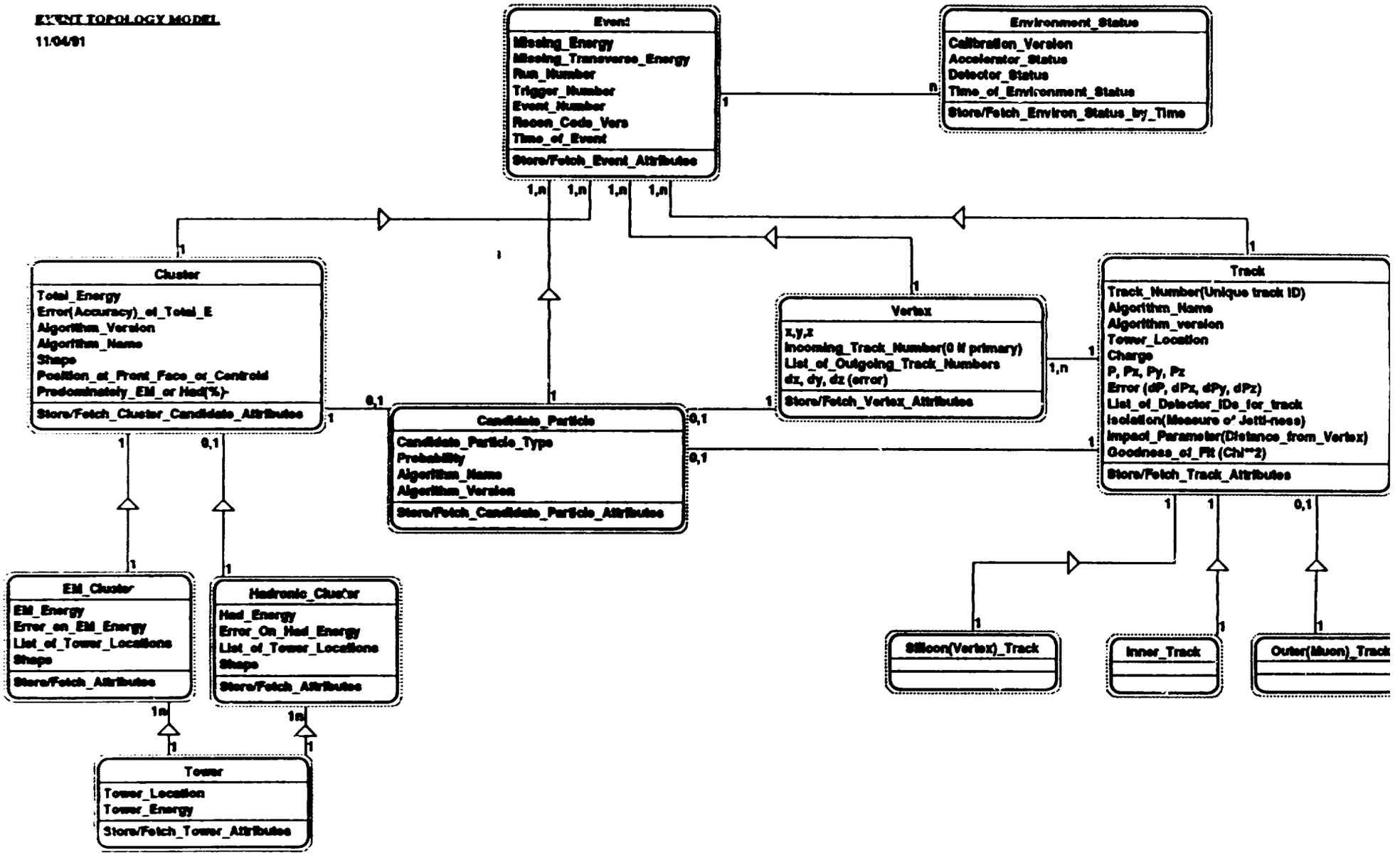

EVENT TOPOLOGY MODEL
11/04/91
Event
Missing_Energy
Missing_Transverse_Energy
Run_Number
Trigger_Number
Event_Number
Recon_Code_Vers
Time_of_Event
Store/Fetch_Event_Attributes
Environment_Status
Calibration_Version
Accelerator_Status
Detector_Status
Time_of_Environment_Status
Store/Fetch_Environ_Status_by_Time
Cluster
Total_Energy
Error(Accuracy)_of_Total_E
Algorithm_Version
Algorithm_Name
Shape
Position_at_Front_Face_or_Centroid
Predominately_EM_or Had(%)
Store/Fetch_Cluster_Candidate_Attributes
Candidate_Particle
Candidate_Particle_Type
Probability
Algorithm_Name
Algorithm_Version
Store/Fetch_Candidate_Particle_Attributes
Vertex
x,y,z
Incoming_Track_Number(0 if primary)
List_of_Outgoing_Track_Numbers
dx, dy, dz (error)
Store/Fetch_Vertex_Attributes
Track
Track_Number(Unique track ID)
Algorithm_Name
Algorithm_version
Tower_Location
Charge
P, Px, Py, Pz
Error (dP, dPx, dPy, dPz)
List_of_Detector_IDs_for_track
Isolation(Measure of Jetti-ness)
Impact_Parameter(Distance_from_Vertex)
Goodness_of_Fit (Chi**2)
Store/Fetch_Track_Attributes
EM_Cluster
EM_Energy
Error_on_EM_Energy
List_of_Tower_Locations
Shape
Store/Fetch_Attributes
Hadronic_Cluster
Had_Energy
Error_On_Had_Energy
List_of_Tower_Locations
Shape
Store/Fetch_Attributes
Tower
Tower_Location
Tower_Energy
Store/Fetch_Tower_Attributes
Silicon(Vertex)_Track
Inner_Track
Outer(Muon)_Track

Figure 8

5. Plans for 1992

5.1 1992 Protoyping and Continued Design

5.1.1 Prototyping

- SDC Simulation Activity by Physicists
  - Realistic Simulation
  - Wide Use by Physicists

- HPCC/DOE Data Base Prototyping by Physicists (Larry Price of Argonne)
  - Relational Version (SyBase) by Ed May of Argonne
  - Object Oriented Version (Objectstore) by Chris Day of Lawrence Berkerley Lab

- IBM FSD Prototyping
  - Prototype a Physics Analysis Framework Consisting of:
    - Object Oriented Data Base Manager (Objectivity)
    - Data Query Rulebase
    - Analysis Subsystem
    - Framework Rulebase
    - User Interface for Physics Analysis
  - Data Migration Via IEEE Mass Storage Reference Model

5.1.2 Continued Software Architecture Analysis & Design

- Continue Object Oriented Analysis on Remainder of Software Architecture

- Use Operational Scenarios to Validate Object Hierarchy

6. References

1. Study Report : Solenoidal Detector Collaboration Computing Section, Architectural Studies and Analysis, December 16, 1991, SDC Computing Working Group and IBM Federal Sector Division

2. Rumbaugh, J. Blaha, M. Premerlani, W. Eddy, F. and Lorensen, W. Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1990.

3. Coad P. and Yourdon, E. Object-Oriented Analysis. Prentice Hall, Englewood Cliffs, NJ, 1990.

4. Cardenas, A. and McLeod, D. Research Foundations In Object-Oriented And Semantic Database Systems. Prentice Hall, Englewood Cliffs, NJ, 1990.

5. Rebecca J. Wirfs-Brock and Ralph E. Johnson, Surveying Current Research In Object-Oriented Design. Communication Of The ACM v.33, n.9, September 1990, pp104-124.

6. Brian Henderson-Sellers and Julian M. Edwards, The Object-Oriented Systems Life Cycle. Communication Of The ACM v.33, n.9, September 1990, pp142-159.

7. Paul T. Ward, How to Integrate Object Orientation with Structured Analysis, IEEE Software, March 1989. Related Class Notes from The CASE Real-Time Curriculum

8. Larry Constantine, The Object Oriented Paradigm, Keynote Address to Object Oriented Systems Symposium, Boston, MA. Summer 1989. Article from American Programmer: Object Oriented and Structured Methods: Toward Integration

9. Sam Schapelle, Susan Lilly, Object Based Ada Software Engineering, Class Notes, July, 1991 (IBM FSD Software Engineering Process)

10. Edward Yourdon, Auld Lang Syne, Byte Magazine, Oct. 1990, pp257-264.

11. Frederick P. Brooks, Jr., Univ. of North Carolina at Chapel Hill, No Silver Bullet, Computer Magazine, Apr. 1997, pp10-19.

Using Object Oriented Analysis & Design To Study the SSCL SDC Computing System

Second International Workshop on Software Engineering, AI, and Expert Systems For High Energy and Nuclear Physics January 13-18, 1992

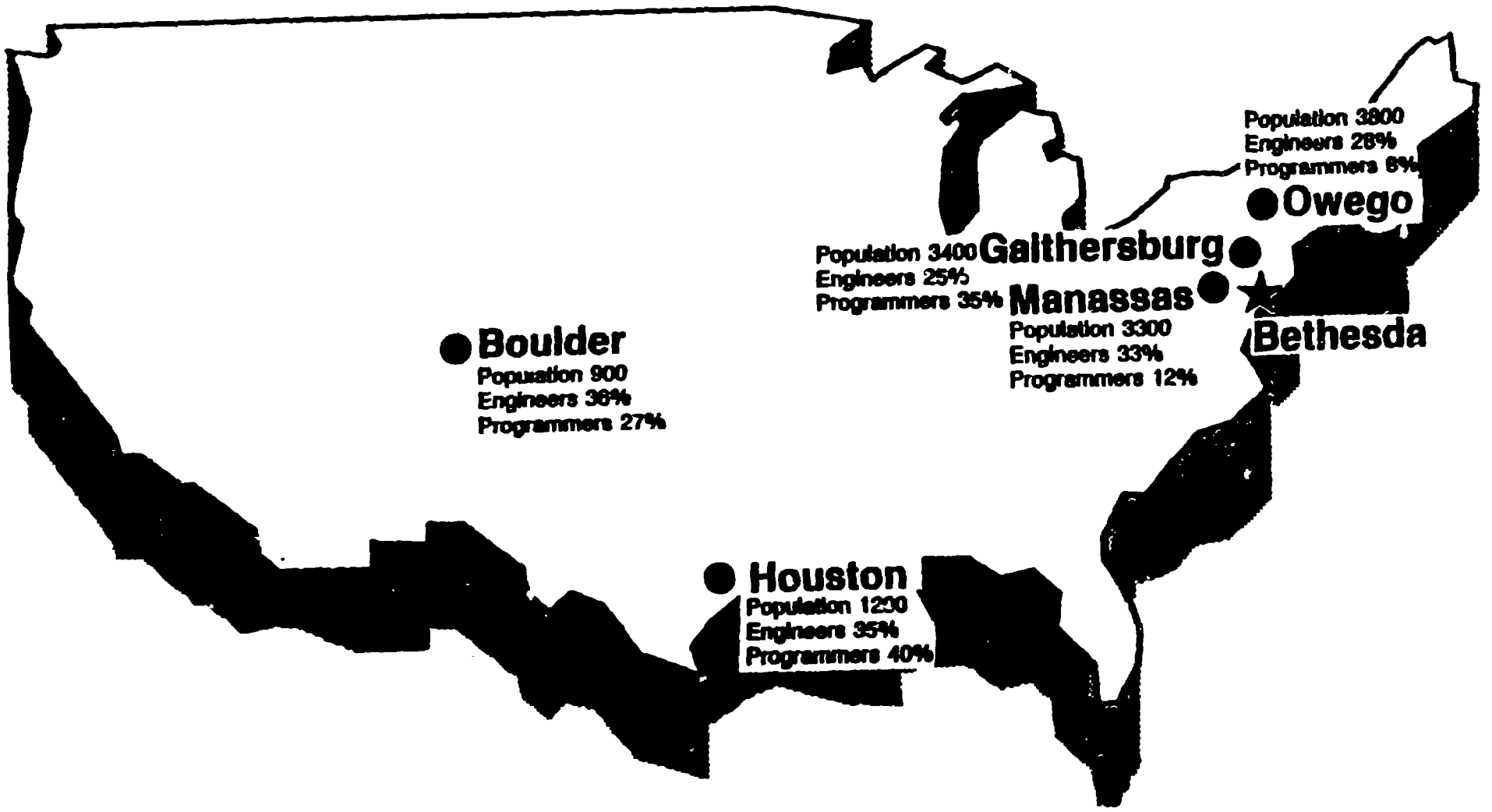
Glenn T. Kubena (IBM), Andrea P.T. Palounek (LANL)
Chris Day (LBL), Ken Liao (IBM), et. al.

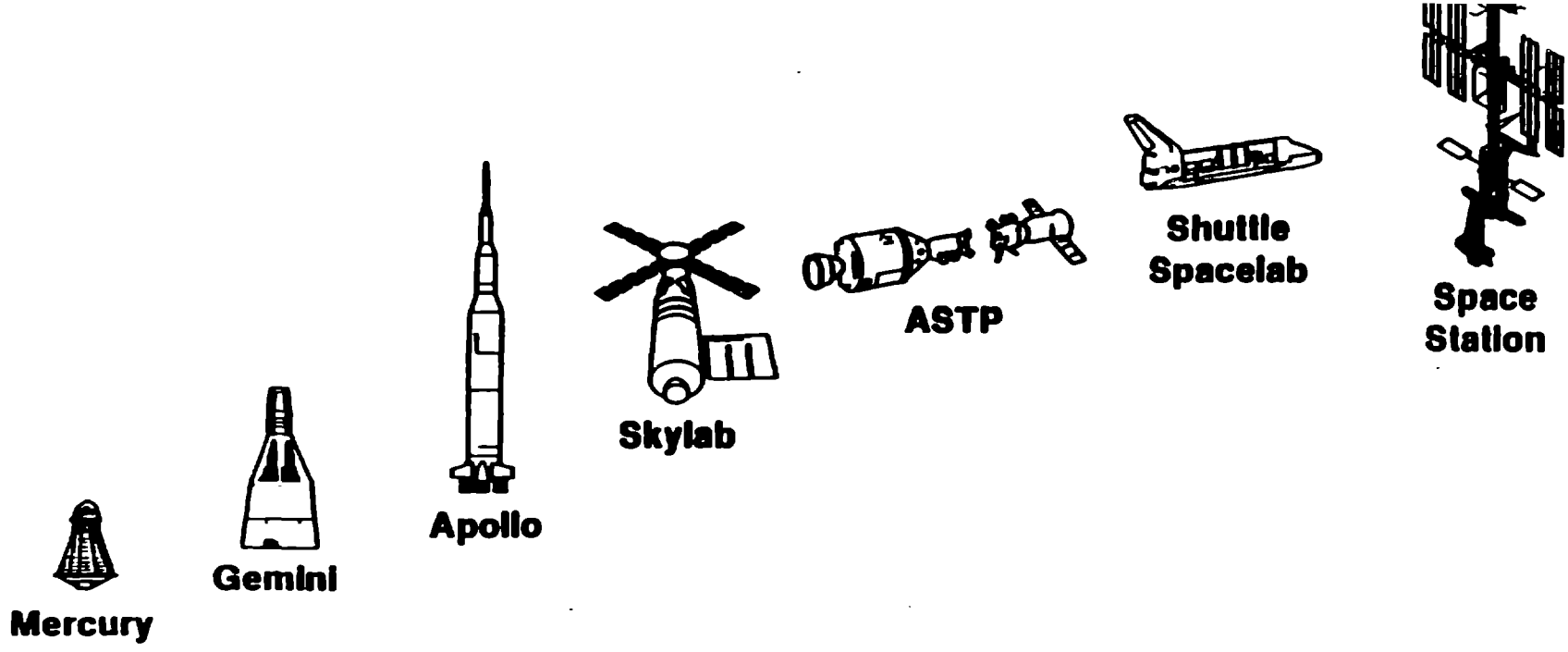
Point of Contact: Glenn Kubena
IBM Federal Sector Division
Houston, Texas
(713) 282-7635
KUBENA@HOUVMSCC.VNET.IBM.COM

Contents

Introduction

FEDERAL SYSTEMS LOCATIONS





Space Station DMS

Flight Computers and Software

Mission Control

1960 1970 1980 1990

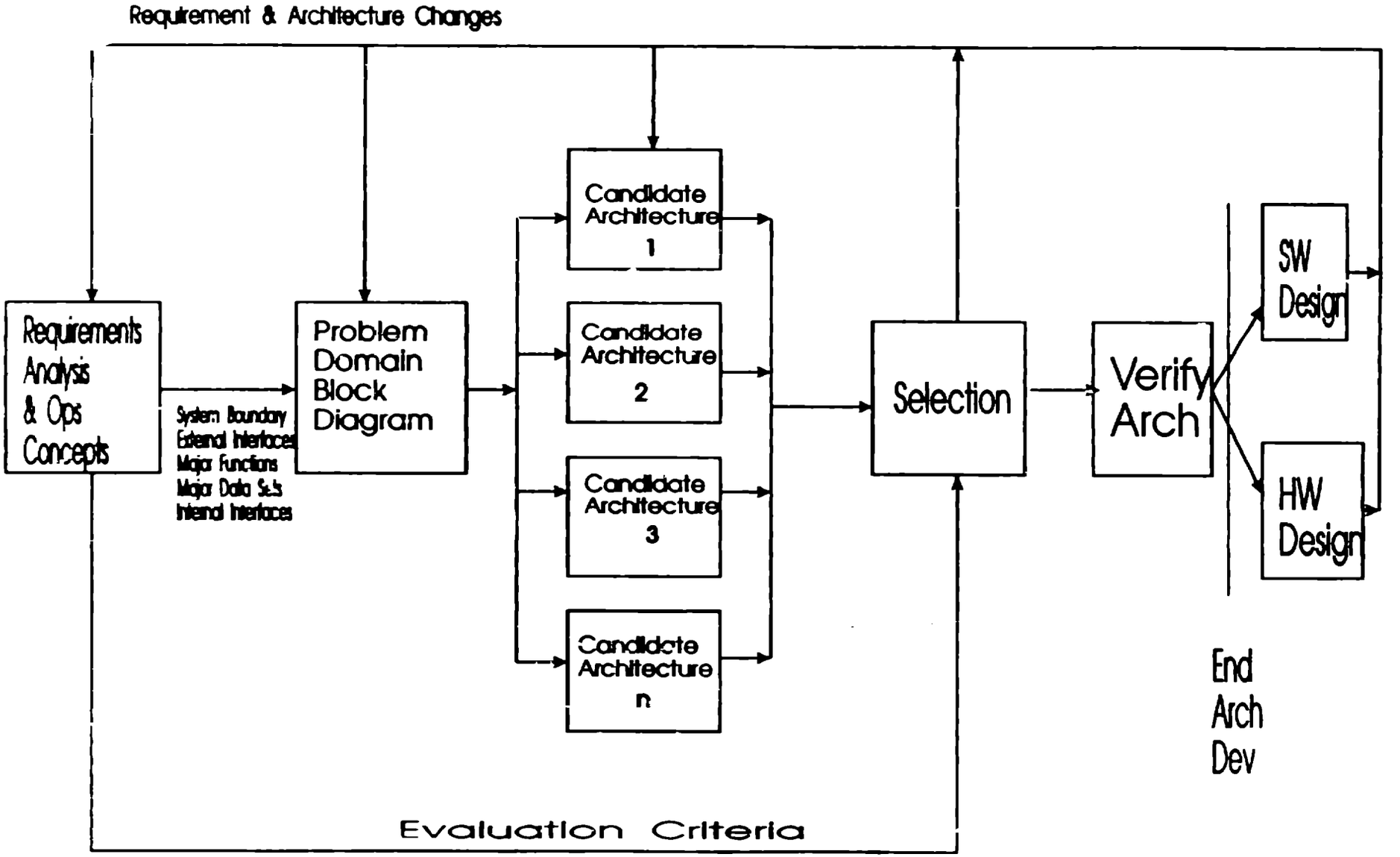
SDC Computing Architecture Study

- Joint Effort Between SDC and IBM FSD
- April to December, 1991
- Early Analysis & Design of SDC Offline Computing System
  - To Support SDC Proposal to Dept. of Energy in April 1992
  - Trade Study Oriented
  - Preliminary Results; Analysis & Design Ongoing
  - Results to Date Documented in Study Report
    - Operational Concepts Development
    - SDC System Architecture Analysis
    - SDC Software Architecture Analysis
    - Performance Modeling
    - Standards and Technology Forecast
    - Costing

- This Presentation Will Cover Results of System & Software Architecture Analysis

SDC System Architecture Analysis

SDC System Architecture Analysis Process Used



Results of SDC System Architecture Analysis

Evaluation Criteria for SDC Offline Computing

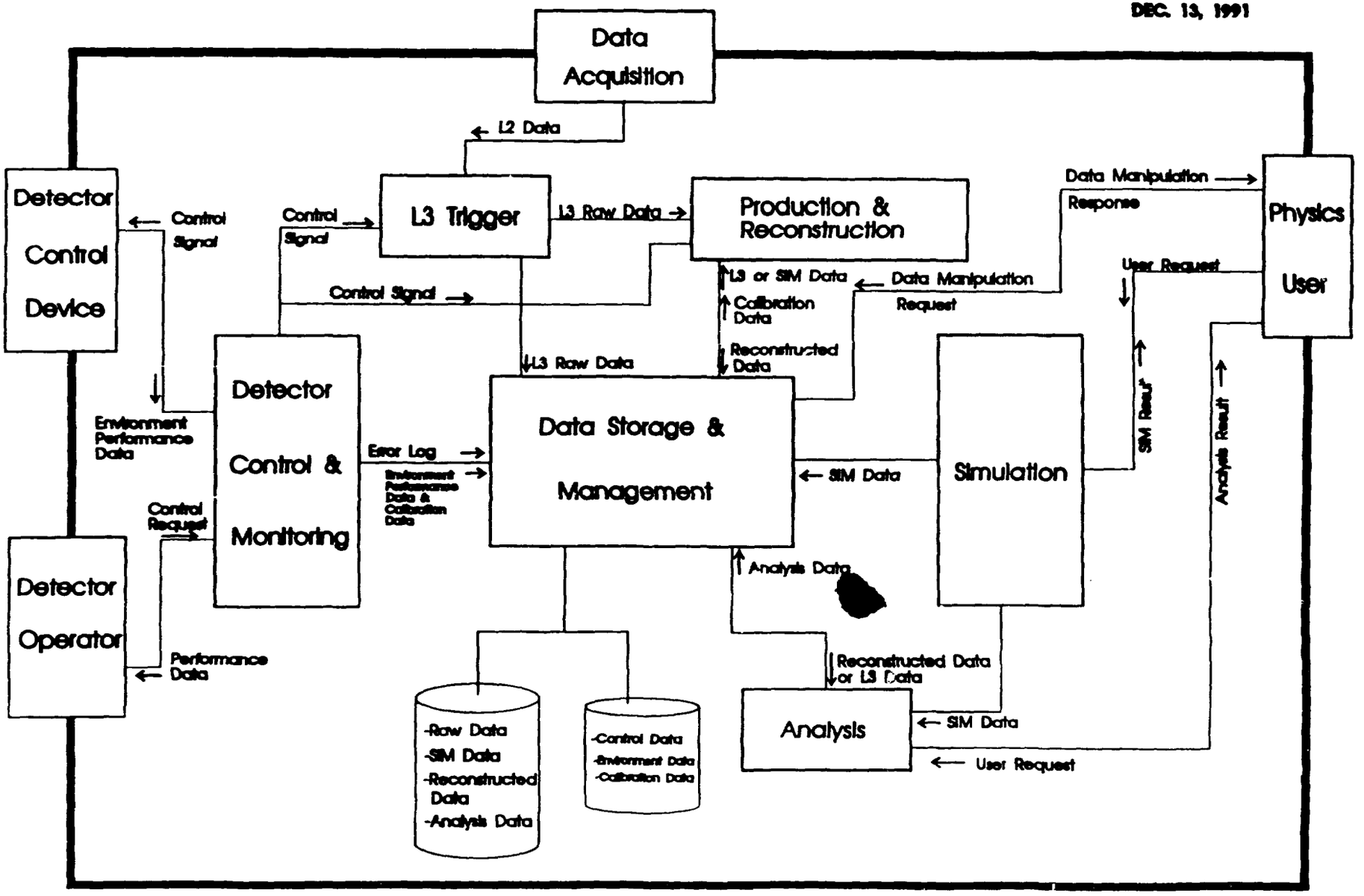
- Cost
  - Hardware acquisition & maintenance costs
  - Software acquisition & maintenance costs (re-use, make or buy options)
    - Public domain (HEPLib, etc.)
    - Custom developed internally or with outside help
    - Commercial software distributed licences & maintenance fees

- Schedule Risk
  - Ability to meet all critical milestones
  - Availability of products to meet evolving standards
  - External dependency risks
  - Availability of critical skills

- Performance
  - 100 raw events/second, received from level 3 processing, processed through production reconstruction without loss of data
  - Storage of 100 reconstructed events/second or 10**15 bytes/year (event size = 1 megabyte; 10**7 seconds/year). An equal amount of storage assumed for raw event data.
  - Any two of three activities supported concurrently (production reconstruction, second pass reconstruction, simulation)
  - Data reduction of the 10**15 bytes of reconstructed data to a 10**14 byte archive sample and a 10**9 DST event online sample.
  - Transfer of a 10**5 event sample extracted from the 10**9 DST event online sample to a single user's workstation within 3 hours.
  - Access to any event in the raw or reconstructed archive within 24 hours. Access to a large number of archived events within 2 weeks.
  - Accommodate 200 concurrent analysis users including 50 at the SSCL and the remainder at regional centers/institutions.

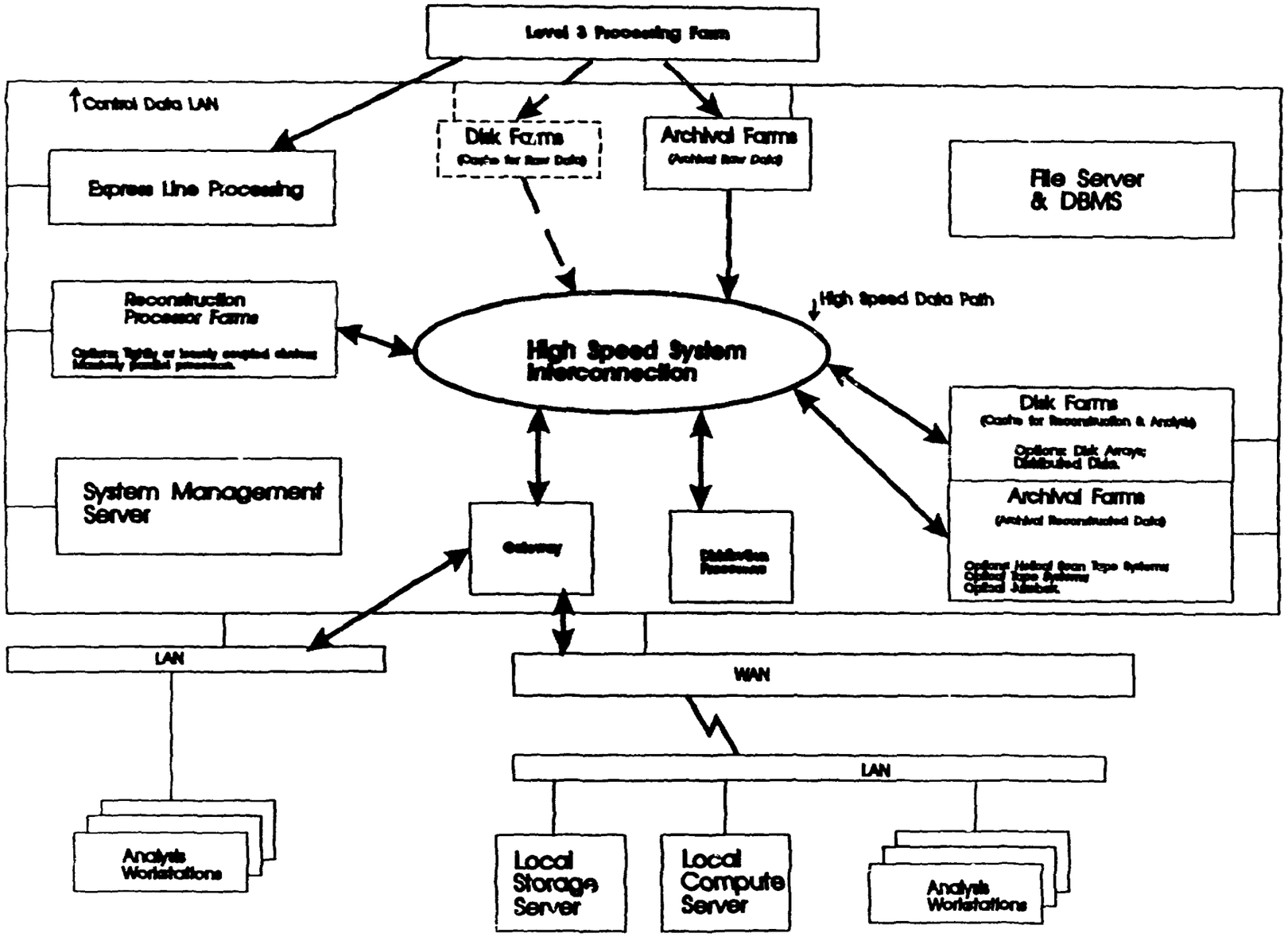
- Environment
  - Technology insertion and software portability
  - Common user interface; support for expert and ad hoc users
  - Data structures transparent to physics analysis users
  - Interoperability between the SSCL and remote institutions
  - Levels of code management across distributed environment, e.g.:
    - Production reconstruction at the SSCL
    - Analysis at remote institutions
  - Automated test tools and test data generation

Problem Domain Block Diagram



SDC Conceptual Architecture Diagram

October 24, 1991



SDC Software Architecture Analysis

Process Used

1. Identified Key Software Architecture Design Goals

2. Proposed a Conceptual Software Architecture Model Consisting of Three Layers

   - System Services Layer - Driven by Portability, Interoperability, Reliability, Common User Interface and Data Transparency Requirements
   - Common Physics Applications Layer - Driven by the Need for a Common Framework for Physics Reconstruction, Analysis, Simulation and Test
   - Unique Physics Applications Layer - Driven by the Need for Easy Analysis Software Development by End Users Within the Common Framework

3. Proposed a Conceptual Physics Analysis Framework

4. Performed Object Oriented Analysis on Data Storage Management

   - Produced Event Topology Model

Results of SDC Software Architecture Analysis

Software Architecture Design Goals

- Software should be maintainable, efficient, reliable, understandable, reusable and extensible.
- Software portability and interoperability should support technology upgrades and multi-vendor distributed environments. The SSCL must provide for standard interoperability with remote institutions. Source code portability of the analysis software environment should be provided for all collaborating institutions.
- The communications software should provide for easy exchange of mail, messages and graphical data between external users.
- The user interface with the software should be standard and easy to use. The user interface should support ad hoc as well as expert users. The user interface should evolve to support point-and-click analysis.
- The software architecture should provide for a software bus or infrastructure of standard services utilizing industry standard components wherever

possible. The goal is to incur the cost of common services software only once and to promote system integrity.

- The software architecture should provide an environment or framework which makes it easy for physicists to add or modify analysis code without relink of the framework. Access to data by such analysis code should be via an easy to use object query language, i.e., knowledge of internal data structures should be transparent to physicists developing routine analysis code.
- The software architecture should include tools for test data generation and validation against predetermined results or test scripts.
- The software architecture should provide a reasonable level of online help, error recovery and error diagnostics.
- The software architecture shall meet all performance requirements.
- Software shall be designed within budget and within schedule.

Software Architecture Model

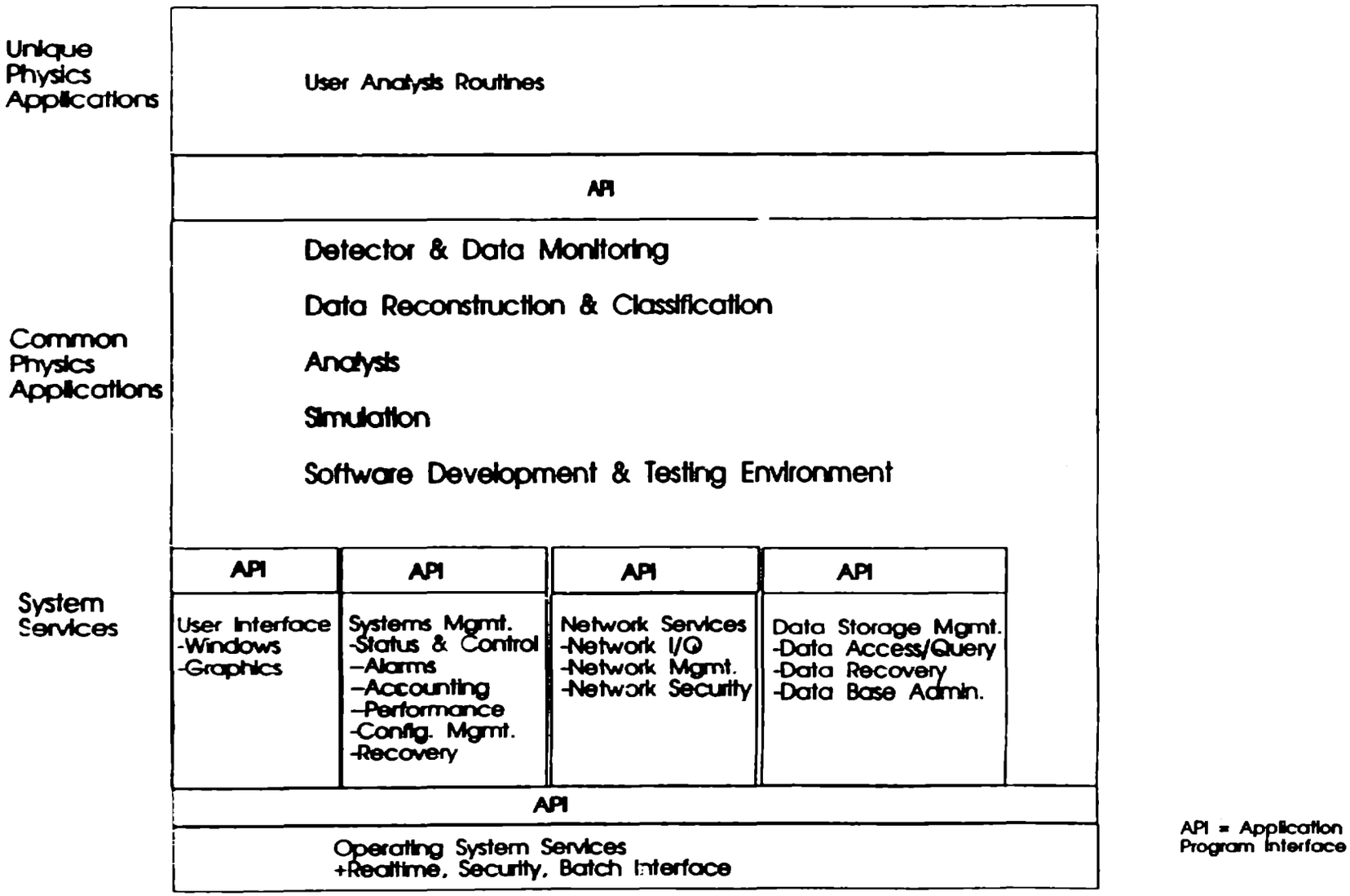
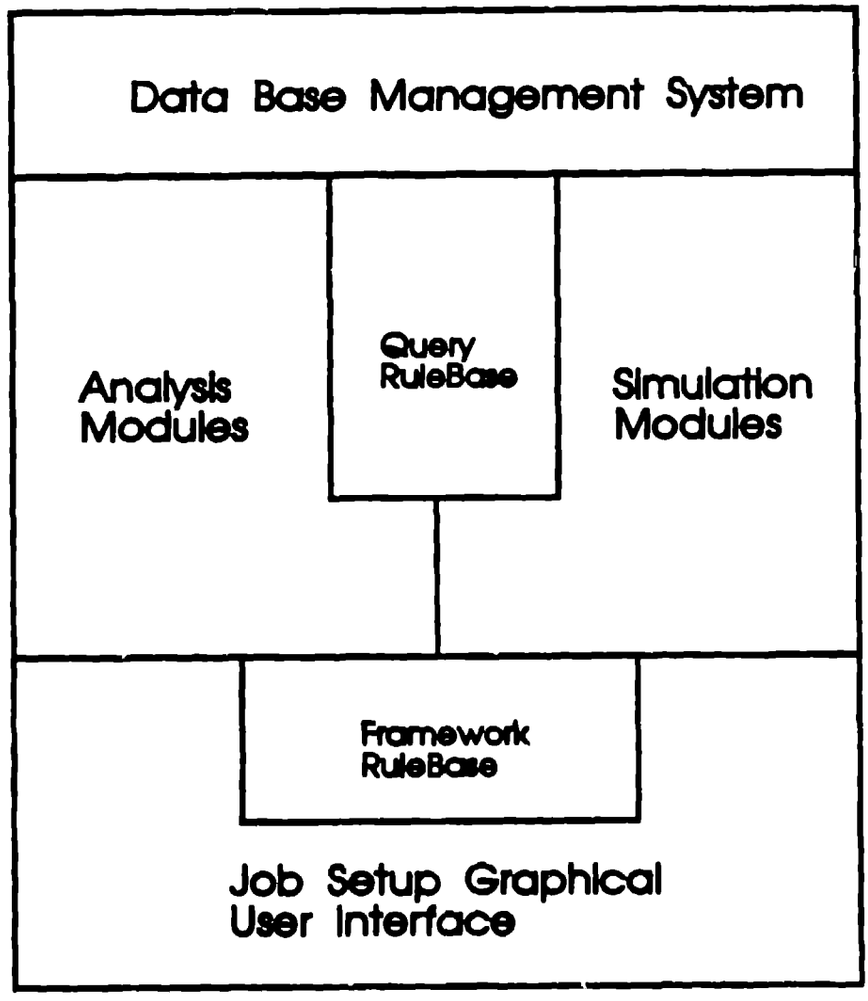


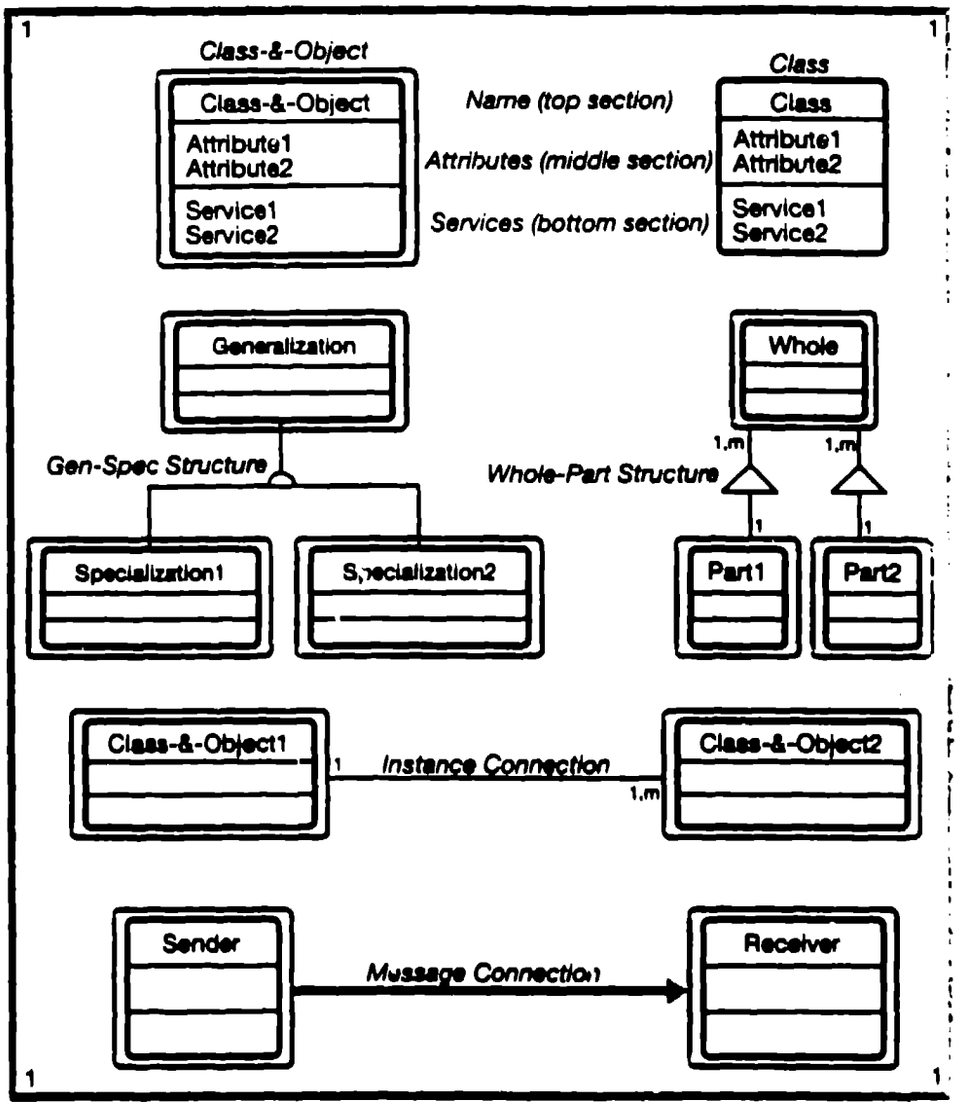
Figure 4-4

Analysis & Simulation Software Architecture

Dec. 10, 1990



Object Oriented Analysis Tool (TM) Notation



Subject (may be expanded or collapsed)

Note: In addition, OOA uses Object State Diagrams and Service Charts for specifying Services.

EVENT TOPOLOGY MODEL

11/04/91

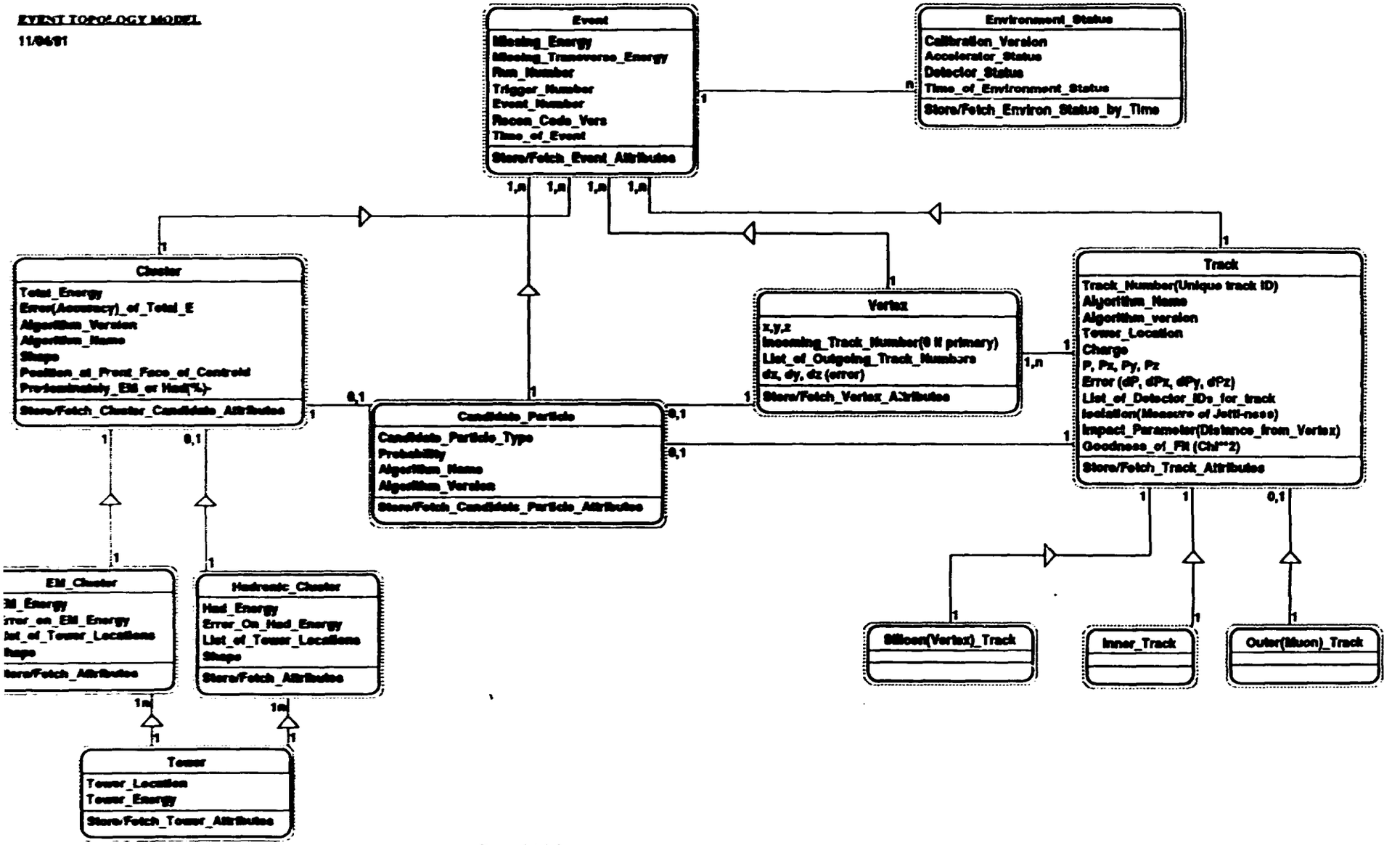


Figure 4.2.1-4

Plans for 1992

1992 Prototyping and On-Going Design

Prototyping

- SDC Simulation Activity by Physicists
  - Realistic Simulation
  - Wide Use by Physicists

- HPCC/DOE Data Base Prototyping by Physicists (Larry Price of Argonne)
  - Relational Version (SyBase) by Ed May of Argonne
  - Object Oriented Version (Objectstore) by Chris Day of Lawrence Berkerley Lab

- IBM FSD Prototyping
  - Prototype a Physics Analysis Framework Consisting of:
    - Object Oriented Data Base Manager (Objectivity)
    - Data Query Rulebase
    - Analysis Subsystem
    - Framework Rulebase
    - User Interface for Physics Analysis
  - Data Migration Via IEEE Mass Storage Reference Model

Continued Software Architecture Analysis & Design

- Continue Object Oriented Analysis on Remainder of Software Architecture

- Use Operational Scenarios to Validate Object Hierarchy

References

1. Study Report : Solenoidal Detector Collaboration Computing Section, Architectural Studies and Analysis, December 16, 1991, SDC Computing Working Group and IBM Federal Sector Division

2. Rumbaugh, J. Blaha, M. Premerlani, W. Eddy, F. and Lorensen, W. Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1990.

3. Coad P. and Yourdon, E. Object-Oriented Analysis. Prentice Hall, Englewood Cliffs, NJ, 1990.

4. Cardenas, A. and McLeod, D. Research Foundations In Object-Oriented And Semantic Database Systems. Prentice Hall, Englewood Cliffs, NJ, 1990.

5. Rebecca J. Wirfs-Brock and Ralph E. Johnson, Surveying Current Research In Object-Oriented Design. Communication Of The ACM v.33, n.9, September 1990, pp104-124.

6. Brian Henderson-Sellers and Julian M. Edwards, The Object-Oriented Systems Life Cycle. Communication Of The ACM v.33, n.9, September 1990, pp142-159.

7. Paul T. Ward, How to Integrate Object Orientation with Structured Analysis, IEEE Software, March 1989. Related Class Notes from The CASE Real-Time Curriculum

8. Larry Constantine, The Object Oriented Paradigm, Keynote Address to Object Oriented Systems Symposium, Boston, MA, Summer 1989. Article from

American Programmer: Object Oriented and Structured Methods: Toward Integration

9. Sam Schapelle, Susan Lilly, Object Based Ada Software Engineering, Class Notes, July, 1991 (IBM FSD Software Engineering Process)

10. Edward Yourdon, Auld Lang Syne, Byte Magazine, Oct. 1990, pp257-264.

11. Frederick P. Brooks, Jr., Univ. of North Carolina at Chapel Hill, No Silver Bullet, Computer Magazine, Apr. 1997, pp10-19.